End-to-End Deep Learning Pipeline for Real-Time Processing of Surface Scattering Data at Synchrotron Facilities

VLADIMIR STAROSTIN, LINUS PITHAN, ALESSANDRO GRECO, VALENTIN MUNTEANU, ALEXANDER GERLACH, ALEXANDER HINDERHOFER, AND FRANK SCHREIBER
Institute of Applied Physics, University of Tübingen, Tübingen, Germany

Introduction

Data analysis is undoubtedly becoming a major bottleneck in experimental science [1, 2]. Therefore, large-scale research facilities such as synchrotrons provide powerful computing infrastructure along with their experimental instruments [3]. Due to the ever-increasing size and acquisition rates of modern area detectors, for many synchrotron users, transferring terabytes of scattering data to the home institutes has become a challenge in itself, not to mention analyzing those data. Due to these enormous data volumes and the desire to make data-driven decisions during the experiment, online data processing and analysis have become key ingredients for many experiments. While computer clusters at synchrotrons advance rapidly and provide users with access to a wide range of computing resources [4], real-time data analysis based on user-developed software usable at different beamlines and facilities remains a challenge.

Surface scattering measurements are no exception to this trend. The high demand for studying thin films for various applications leads to huge amounts of measured data that require smart solutions for automated analysis. Recently, several deep learning approaches have been presented for accelerating the analysis of surface scattering data in specular [5, 6] and off-specular [7] geometries. In this way, the specular reflectometry data can be processed via an open-source machine learning-based software [8]. Here, we focus on two-dimensional off-specular data obtained by grazing-incidence X-ray diffraction (GIXD) measurements; however, the concept of a user-driven and facility-independent end-to-end pipeline is not limited to this geometry.

GIXD is an indispensable technique for studying crystalline structures on surfaces [9] and it allows, *inter alia*, real-time tracking of crystallization processes, which is crucial for a wide array of applications. There is a high demand for automated solutions for fast GIXD data processing due to large acquisition rates and time-consuming data analysis. For instance, one typically obtains around 100,000 diffraction images (≈ 2 terabytes) per beam day of real-time measurements, and these numbers are expected to increase further due to continuously improving experimental techniques and scientific demands. The fast analysis of GIXD data requires computationally expensive preprocessing of images as well as GPU-accelerated deep learning algorithms for peak identification, as described in Ref. [7]. To enable the real-time use of this pipeline across different synchrotron facilities, the corresponding software should be easy to integrate into the IT infrastructure of different facilities [10, 11]. Usually, such an integration requires the involvement of computing staff at the facility with deep background knowledge of the respective IT infrastructure. In turn, this may lead to highly optimized software solutions for one specific instrument or facility where the portability of the entire data handling pipeline is not of key interest. For certain non-routine experiments or experiments of IT-affiliate user groups, there is the evident need for data handling pipelines that are not bound to individual facilities or instruments and offer a high degree of flexibility to be deployed easily elsewhere. In this context, the best approach seems to be a facility-independent software that allows optional, facility-specific integration for further optimization. Such an approach additionally promotes and highlights the importance of the standardization of data handling, processing, and storage among the different facilities, user groups, and scientific communities.

In this work, we discuss challenges and possible approaches for building a GIXD data processing pipeline. We demonstrate the implementation of such a software framework using *gixi* (Grazing Incidence X-ray diffraction Intelligent pipeline), an open-source package based on the deep learning approach introduced in Ref. [7] that provides an end-to-end solution for automated GIXD analysis, including image processing, detection of the diffraction peaks, peak intensity extraction, and crystal structure identification with the focus on powder diffraction. Our user-designed implementation allows a straightforward integration into any cluster infrastructure based on the commonly used Slurm Workload Manager.

So far, the package has been successfully employed for processing large amounts of GIXD data at two different synchrotron facilities: (1) the PETRA III X-ray radiation source of the Deutsches Elektronen-Synchrotron (DESY); and (2) the European Synchrotron Radiation Facility (ESRF). We emphasize, however, that the package can be effortlessly adapted for use at other facilities. The modular structure of the software allows further facility-specific integration, especially direct connections to detector data streams that do not involve disk storage. Furthermore, we discuss the main concepts of the standardized reports on data analysis according to the guidelines of the DAPHNE4NFDI projects and illustrate them by publishing an exemplary dataset with *gixi*-analyzed *in situ* GIXD measurements of perovskite crystallization.

The pipeline for GIXD data analysis

Deep learning is a promising choice for analyzing complex 2D scattering data with various experimental artifacts and diffraction features. In general, GIXD images contain rich and diverse information about the studied sample. Based on the concrete scientific problem, different types of analysis are required to extract the relevant properties of the studied system. These may include the lattice parameters, the texture, and fractions of co-existing mixtures in case of powder diffraction, time-dependent properties of the studied process in case of *in situ* measurements. Due to the large number of related quantities, it can be inefficient to develop separate software solutions for each type of analysis. Since the diffraction peaks contain the most relevant information about the sample in wide-angle geometry, the natural approach is to separate the analysis into an application-agnostic detection of diffraction peaks that determines their positions, sizes, intensities, and other relevant characteristics followed by an application-specific analysis step that relies on the detection results.

We illustrate this approach with the *gixi* package, which is written in Python and comprises a server-side and a client-side application that can be used separately. The server provides all the image processing and data analysis operations from preprocessing the raw data to peak detection and saving the results. The client is a lightweight PyQt5-based graphical user interface (GUI) designed for visualizing both the raw images and the processed results, as well as for starting the server jobs on a cluster for real-time image processing. The server operations require a fast implementation with multiprocessing and CUDA support to enable real-time analysis.

In the following, we discuss the image processing steps required for GIXD data analysis and how they are implemented in our package.

Fast image preprocessing

The raw diffraction images obtained from the detector require certain preprocessing steps before they can be analyzed by the peak detection algorithm. The first step involves an image conversion to reciprocal space. The shape of the processed diffraction peaks is critical for deep learning detection algorithms and the underlying architecture to function properly. To be able to use well-established detection algorithms for our task, a diffraction peak in the processed image must have shape and position that can be characterized by a rectangular bounding box. For single-crystal diffraction, this condition is usually already satisfied in reciprocal space $\{q_{xy}, q_z\}$. In contrast, for powder diffraction, the radial symmetry usually requires a conversion to polar reciprocal coordinates $\{\|q\| \equiv q_{xy}^2 + q_z^2; \phi \equiv \arctan(q_z/q_{xy})\}$ to obtain rectangular features (see Figure 1c). Therefore, an analysis pipeline should provide adequate image conversions that depend on the scattering geometry used.

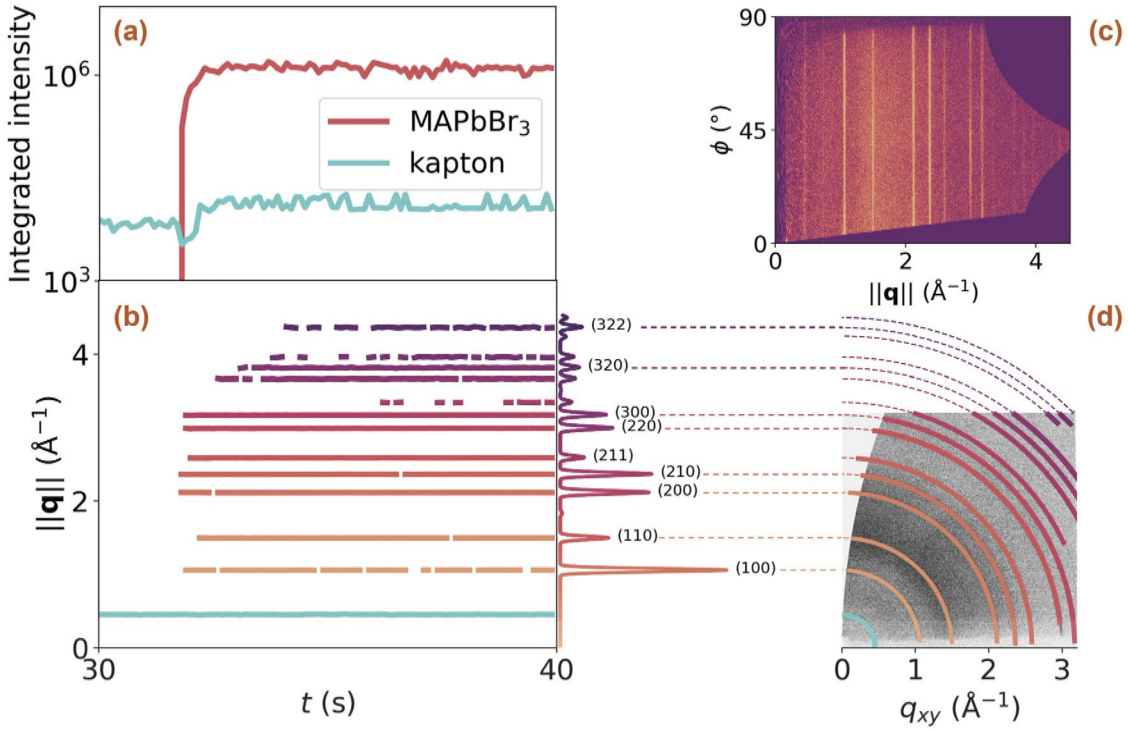

Figure 1: Results of the automated analysis of the published dataset. (a,b) Illustrate time dependencies for profiles of integrated intensities and peak positions ‖**q**‖ *correspondingly. The detected peaks are matched against the simulated diffraction profile of* $MAPbBr_3$ *shown on the right-hand side of (b). (c) The processed image in polar reciprocal coordinates with enhanced contrast. (d) The processed image in reciprocal coordinates with detected peaks.*

In general, the conversion operation is computationally expensive. For instance, the histogram-based Cython implementation in the pyFAI package [12] reaches 30 Mpix/s, which corresponds to roughly ≈ 7 images/s for a detector with 4 Mpix resolution. To reach real-time data processing capabilities for higher acquisition rates, we overcome this bottleneck in the *gixi* implementation with a tenfold acceleration on a single processor by using the interpolation algorithm from the OpenCV library. To achieve this, the analytical dependencies $y(\|\mathbf{q}\|, \phi)$, $z(\|\mathbf{q}\|, \phi)$, $y(q_{xy}, q_z)$, and $z(q_{xy}, q_z)$ required for the interpolations are derived, where $\{y, z\}$ are the detector coordinates.

As the second preprocessing step, the contrast of a diffraction image is corrected to reduce its dynamic range and optimize for further analysis with the neural network. For contrast correction, we use a fast OpenCV implementation of contrast-limited adaptive histogram equalization (CLAHE). We note that the intensity correction is only required for the detection step and the original intensity levels should be used for any further analysis.

For both interpolation and contrast enhancement, we find CPU implementations by OpenCV sufficient for our tests, but one can further accelerate them via PyTorch-based CUDA implementation of the interpolation algorithm if required by the experimental setup. Furthermore, the package provides optional multiprocessing acceleration for the image preprocessing step (see Figure 2).

Deep learning-based peak detection

Presently, there are multiple competing deep learning object detection techniques that could be employed for the peak detection task. In *gixi*, the detection model is based on the Faster R-CNN architecture [13] with several modifications that make it lightweight and more accurate compared to the standard Faster R-CNN as well as to some other popular architectures [7].

We use a convolutional neural network based on ResNet-18 architecture as a feature extractor to convert a preprocessed image in polar reciprocal coordinates to three sets of asymmetric feature maps at three different scales. The asymmetric strides of the used convolutional layers ensure minimal compression of a diffraction image along the horizontal axis to preserve the high $\|\mathbf{q}\|$ resolution required for the accurate determination of $\|\mathbf{q}_{peak}\|$ positions. At the same time, the images are compressed in angular dimension (×8, ×16, and ×32 compression for each feature map, respectively). Larger feature maps are suited for the detection of small diffraction spots and the most compressed maps are aimed at extracting prolonged features such as Debye–Scherrer rings in powder diffraction images. The corresponding convolution-based compression operation amplifies weak peaks and filters out experimental artifacts, as opposed to standard radial profile integration.

The obtained feature maps are then used by the Region Proposal Network (RPN) to provide regions of interests (ROI) that are then refined by the second detection stage, which outputs the positions of the detected diffraction peaks along with the confidence scores. All of the corresponding networks are optimized and reduced in size to enable fast inference and low memory consumption. For a more detailed description of the model architecture, the simulation of the training data, and the training process, we refer to Ref. [7]. The model is implemented in the PyTorch framework [14]. We note that PyTorch allows Just-In-Time (JIT) compilation of a neural network into a C++ compatible TorchScript program for reducing inference time. At the moment, JIT is not used in our pipeline since the inference time is sufficiently small and suited for the used acquisition rates (≈ 120 images/s for 4 Mpix res-

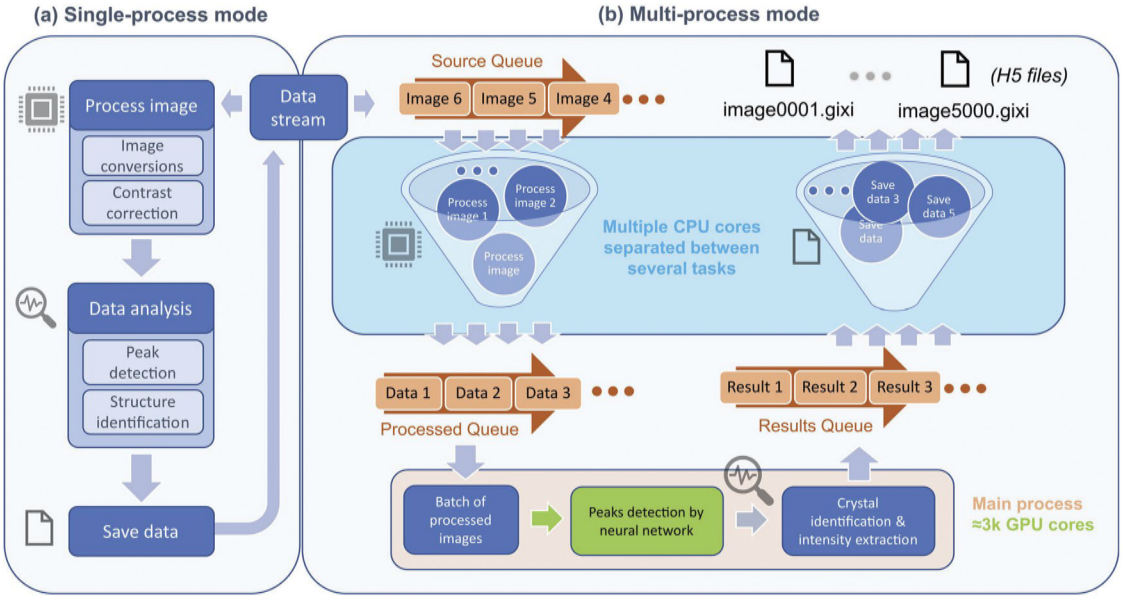

Figure 2: A schematic illustration of (a) the single-process and (b) the multi-process pipeline modes of the server-side application.

olution on NVIDIA GeForce RTX 2080 Ti [7]), but it can be included without much effort for more demanding conditions.

Crystal structure identification

As discussed earlier, an analysis pipeline should be designed in a way that allows application-specific extensions for certain types of analysis. Here, we focus on powder diffraction and provide a corresponding solution for crystal structure identification based on the obtained peak positions. However, we note that the pipeline is in principle extendable to analysis settings and scattering geometries.

GIXD powder diffraction patterns are in general ambiguous and require some degree of additional information about the expected crystal structures to perform a complete structure identification. Apart from the phase problem, possible sources of ambiguity could be, among others, an insufficiently large q range, co-existing powder mixtures, a low signal-to-noise ratio, and a strong scattering background. In our pipeline, structure identification is implemented via a matching algorithm [7] that compares the obtained diffraction peak positions against those from a set of crystal structures provided by the user in the form of Crystallographic Information Files (CIF). The algorithm calculates the probability that a given crystal structure is identified and assigns each peak with the closest Miller index. Figure 1b illustrates the results of the matching algorithm. The current CPU-implementation of the matching operation takes less than 1 ms.

Saving the results

The processed data is stored as hdf5 file that can be accessed by the client program to visualize the results. Typically, this operation takes less than 1 ms. Each file includes the configuration parameters including the experimental geometry, the processed image in polar **q** coordinates (and optionally in standard reciprocal coordinates $\{q_{xy}, q_z\}$), the obtained positions, confidence scores and intensities of the detected peaks, and the matching results. Following the suggested practices of processing and storing the scattering data by DAPHNE4NFDI [15, 16], we intend to provide a standardized NeXus format for processed and analyzed GIXD data in the future.

Multiprocessing implementation

Most synchrotron facilities provide users with powerful computer clusters and the corresponding IT infrastructure to accelerate data processing and simplify data storage. Clusters allow massive parallelization of demanding processing procedures via multiple CPU cores, GPUs, and multiple cluster nodes. *gixi* supports CUDA through PyTorch and optional multiprocessing to accelerate the data pipeline. Figure 2 illustrates both the simple single-process scheme suited for usage on a local computer and the multiprocessing schemes for a cluster node implemented on the server side. The job is submitted via the Slurm Workload Manager, which is currently a common choice for computer clusters.

Client side

The GUI client provides the functionality to access and visualize the data, set the configuration parameters, and submit jobs to the cluster. Figure 3 illustrates the main window of the current version of the GUI application that includes interactive file viewer, image viewer, and logs from the server. It also supports automated periodic fetching of the incoming data for real-time visualization of the data analysis from the cluster along with the cluster logs. To submit a job, the user has to fill in the configuration settings that include data sources, cluster options, and parameters of the experimental geometry. Optionally, some additional settings can be provided, such as postprocessing parameters, the choice of pre-trained detection model, logging options, etc. Our intention is to maintain and extend the client functionality in the future to enable such features as the visualization of the intensity profile and the peak position changes in time, visualization of the matching results, etc.

Multi-facility tests

Our solution requires only a Slurm-based cluster environment and it has been designed having inter-facility portability in mind, since it has been developed by users without a certain facility affiliation. The modular structure of the implementation enables some facility-specific ad-

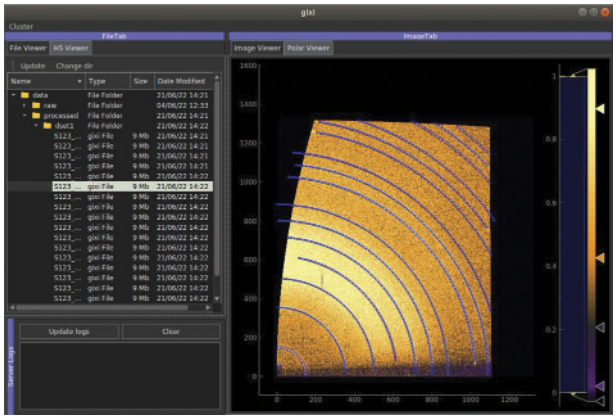

Figure 3: A screenshot of the main window of the client-side application that includes the file viewer, the image viewer, and the cluster logs.

justments if such are required for further optimization. Thus, the most important optimization, which is practically unfeasible to implement in general fashion, is fast data stream handling. By default, our implementation relies on fetching incoming data from a disk storage. However, other schemes that bypass the disk storage should be preferred for higher acquisition rates. These direct data stream connections can be implemented for specific detectors, beamlines, and IT infrastructures without any other modifications to the pipeline. In this way, we hope that our approach will serve as an example of user-designed software for fast data analysis and inspire other research groups to contribute to the standardization of processing and storing of different types of scattering data.

So far, the *gixi* package has been employed for GIXD data analysis for multiple *in situ* experiments at PETRA III with real-time data processing on the Maxwell HPC cluster at DESY [17]. Furthermore, the package has been successfully tested on the Networked Interactive Computing Environment (NICE) provided by the ESRF. In total, more than 1M diffraction images have been processed during the tests performed so far.

As an illustration of the workflow of the pipeline, we publish an exemplary analyzed dataset measured at PETRA III and processed automatically by *gixi* on the Maxwell cluster. The measured diffraction images reveal crystallization of methylammonium lead bromide perovskite ($MAPbBr_3$) by applying an isopropanol antisolvent during spin-coating of the precursor solution. The full description of the experimental setup is provided with the raw data in Zenodo repository [18], and Figure 1 illustrates the results obtained by *gixi*. The detected peaks from each diffraction image are matched against the expected $MAPbBr_3$ crystal structure. The time-resolved peak positions from $MAPbBr_3$ and the corresponding integrated intensity profile appear at $t \approx 32$ s and remain stable afterward. Given that it takes ~2 s with the used setup for antisolvent to be fully dispensed after the command is sent at $t = 30$ s, it is evident that the crystallization process of 3D $MAPbBr_3$ perovskite is almost instantaneous after isopropanol antisolvent is dispensed in the considered time scale.

Raw, processed, and analyzed data are published in separate repositories. Table 1 summarizes the information about published repositories, and the related details are discussed in the next section.

Open science and research data management

Following the broad consensus in the scientific community to establish FAIR research data management, this publication is meant to provide a first blueprint that adopts the DAPHNE4NFDI vision in terms of data accessibility. DAta from PHoton and Neutron Experiments for NFDI (DAPHNE4NFDI) [15, 16] is an NFDI [26] consortium funded by the German Research Foundation (DFG) that tightly connects to the European projects PaNOSC and ExPaNDS. DAPHNE4NFDI engages directly with the user community to develop user-driven data solutions to advance scientific experiments.

With respect to open data, DAPHNE4NFDI—similar to other open data initiatives—proposes to establish a transparent and traceable chain of all steps from the raw data to the final peer-reviewed scientific publication [27]. This is implemented here while fully relying on already existing infrastructure, as summarized in Figure 4. For all data and software entries in the chain, individual DOIs are minted that connect related items. The current implementation relies fully on the Zenodo service [28]. However, in the future, the raw data will most likely be provided through the data catalogue of the facility where the data has been acquired. Furthermore, also for the following steps in the chain, there may be dedicated services arising in the future. Through the bi-directional links (blue arrows in Figure 4), all data and its usage can be traced from the raw data to the final publication and vice versa. In addition, the whole chain is referenced in the final publication (orange arrows). It is an important aspect of DAPHNE4NFDI to ensure the re-

Table 1: The references to the repositories with the published data and the used software for data processing and analysis.

Published data	**Repositories**	**Description**
Raw data	Zenodo [18]	The original detector images of the GIXD measurement together with the corresponding metadata
Processing software	Zenodo [19], GitHub [20]	Conversion of the detector images to polar reciprocal space (a part of *gixi*)
Processed data	Zenodo [21]	Diffraction images converted to reciprocal and polar reciprocal space
Analysis software	Zenodo [22], GitHub [23], PyPi [24]	The *gixi* package
Analyzed data	Zenodo [25]	Dataset containing detected diffraction features and matched peak indices for provided time-resolved data
Due to the current lack of standardized publishing workflows, we manually prepared and inter-linked the following data and software resources via widely used repositories such as Zenodo, GitHub, and PyPi.		

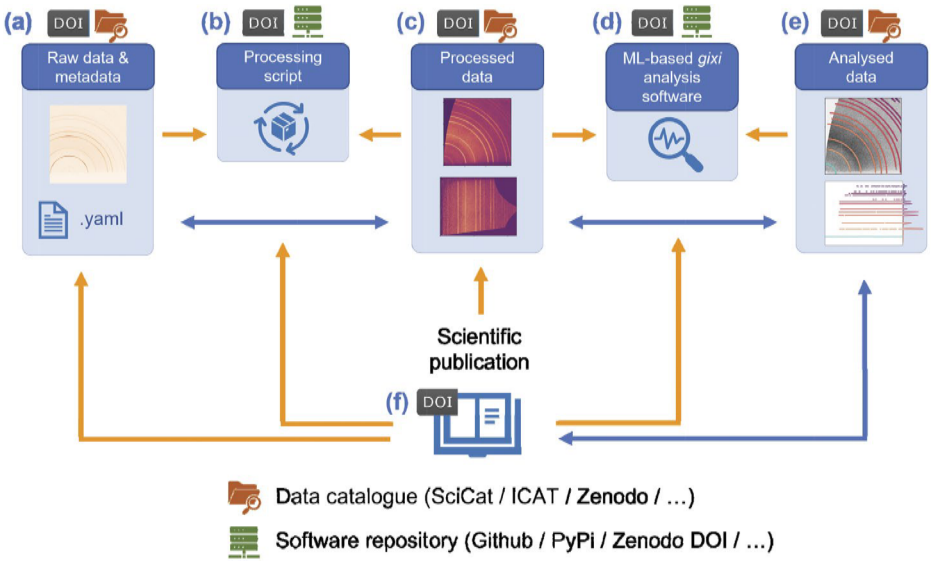

Figure 4: Scheme of the bidirectionally linked open-data publication chain. (a) Raw data collected at facility, including necessary meta-data (e.g., calibration data). (b) Transformation to reciprocal space (a part of the gixi package that is also published separately in line with the guidelines of the DAPHNE4NDFI project). (c) Images converted to q-space are stored to h5 files. (d) gixi software package (feature detection, intensities extraction, crystal structure identification, etc.). (e) Analyzed data stored to h5 files. (f) Final publication. For all the entries in the chain, corresponding DOIs are minted. Blue arrows correspond to bidirectional links and orange arrows represent unidirectional links.

usability of the provided software and intermediate datasets via standardization of the corresponding data and metadata formats as well as of the software environments. This is achieved, e.g., through the contribution of suitable NeXus definition [29] and run-time environments with readily deployed power user software.

Conclusion and outlook

In this work, we discussed the benefits and challenges of creating user-developed end-to-end pipelines for fast and automized data analysis at synchrotron facilities. This was demonstrated with the implementation of *gixi*, an open-source package for GIXD data processing and analysis including deep learning-based detection of diffraction peaks and crystal structure identification for powder diffraction. We considered the possible extension of the presented approach on different geometries and types of analysis.

We also discussed aspects of open data, such as the need for a further standardization of data publishing workflows and data formats related to the publishing chain. Especially in the light of machine learning, which relies on the availability of high-quality reference data for training and testing, the importance of accessible, curated datasets is increasing. Hence, we published [18, 21, 25] the data processed and analyzed with *gixi* following the DAPHNE4NFDI guidelines for a transparent publication chain. We have shown that these requirements can already be met with standard solutions and we encourage other research groups to follow these guidelines and provide more data to the public domain to improve machine learning algorithms for more accurate automatic data analysis.

Acknowledgments

We acknowledge DESY (Hamburg, Germany), a member of the Helmholtz Association HGF, for the provision of experimental facilities. Parts of this research were carried out at PETRA III and we would like to thank Chen Shen, Florian Bertram and Rene Kirchhof for assistance in using the beamline P08. The beamtime was allocated for proposal II-20190761. This research was also supported in part through the Maxwell computational resources operated at DESY with the assistance of Andre Rothkirch and Frank Schlünzen. We also thank the European Synchrotron Radiation Facility (ESRF) for providing access to the Networked Interactive Computing Environment (NICE) cluster.

Funding

This research is part of a project (number: 05K19VTC) funded by the German Federal Ministry for Science and Education (BMBF). ■

References

1. A. Heiss, *Comput. Softw. Big Sci.* **3** (1) (2019). doi:10.1007/s41781-019-0030-7
2. H. Dong et al., *NPJ Comput. Mater.* **7** (1), 1 (2021). doi:10.1038/s41524-021-00542-4
3. C. Wang et al., *Small* **14** (46), 1802291 (2018). doi:10.1002/smll.201802291
4. J. Reppin et al., *Comput. Softw. Big Sci.* **5** (1) (2021). doi:10.1007/s41781-021-00058-y
5. A. Greco et al., *J. Appl. Crystallogr.* **52** (6), 1342 (2019). doi:10.1107/S1600576719013311
6. A. Greco et al., *Mach. Learn: Sci. Technol.* **2** (4), 045003 (2021). doi:10.1088/2632-2153/abf9b1
7. V. Starostin et al., *NPJ Comput. Mater.* **8** (1) (2022). doi:10.1038/s41524-022-00778-8

8. A. Greco et al., *J. Appl. Crystallogr.* **55** (2), 362 (2022). doi:10.1107/S1600576722002230
9. S. K. Sinha et al., *Phys. Rev. B.* **38** (4), 2297 (1988). doi:10.1103/PhysRevB.38.2297
10. S. Roobol et al., *J. Appl. Crystallogr.* **48** (4), 1324 (2015). doi:10.1107/S1600576715009607
11. M. Langer et al., *J. Synchrotron Radiat.* **28** (4), 1261 (2021). doi:10.1107/S1600577521004951
12. G. Ashiotis et al., *J. Appl. Crystallogr.* **48** (2), 510 (2015). doi:10.1107/S1600576715004306
13. S. Ren,C. Cortes, N. Lawrence, D. Lee, M. Sugiyama and R. Garnett, editors. Faster R-CNN: Towards real-time object detection with region proposal networks in *Advances in Neural Information Processing Systems* (Curran Associates, Red Hook, NY, 2015), Vol. 28, p. 91–99.
14. A. Paszke et al., Pytorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems*, edited by H. Wallach et al. (Curran Associates, Inc., Red Hook, NY, 2019), Vol. 32, p. 8024–8035.
15. DAPHNE website: https://www.daphne4nfdi.de/.
16. DAPHNE proposal: https://www.daphne4nfdi.de/downloads/Daphne_proposal.pdf.
17. C. Beyer et al., *EPJ Web Conf.* **245**, 07036 (2020). doi:10.1051/epjconf/202024507036
18. Dataset (raw data). https://doi.org/10.5281/zenodo.6683617
19. Software (processing script): https://doi.org/10.5281/zenodo.7015700
20. Software (processing script): https://github.com/schreiber-lab/fastxq
21. Dataset (processed data): https://doi.org/10.5281/zenodo.6683626
22. Software (analysis): https://doi.org/10.5281/zenodo.7015698
23. Software (analysis): https://github.com/schreiber-lab/gixi
24. Software (analysis): https://pypi.org/project/gixi/0.0.1/
25. Dataset (analysed data): https://doi.org/10.5281/zenodo.6683659
26. NFDI website: https://www.nfdi.de.
27. L. Beddrich et al., arXiv Preprint (2020). arXiv:2010.12086 doi:10.48550/arXiv.2010.12086
28. Zenodo open repository 2015 https://www.zenodo.org.
29. M. Könnecke et al., *J. Appl. Crystallogr.* **48**, 301 (2015).